AMENDMENTS TO THE CLAIMS

Listing of Claims:

1. (Original) A method for out of order read or write scheduling using at least one queue, comprising:

sorting memory transactions into at least one queue;

within at least one queue, sorting memory transactions by attributes;

scheduling memory transactions out of order in accordance with scheduling algorithms, wherein if a DRAM command of a later memory transaction is scheduled before the last DRAM command of an earlier memory transaction, the later memory transaction is considered scheduled out of order and wherein the entire at least one queue is considered while making scheduling decisions;

making out of order selection at the time of DRAM command launch.

2. (Original) The method claimed in claim 1, wherein making out of order selection at the time of DRAM command launch further comprises:

in response to making an out of order selection at the time of DRAM command launch and not earlier, allowing at least one queue to fill up without delaying the DRAM command.

3. (Original) The method claimed in claim 1, further comprising:

accommodating page closing policy with no prior assumption being made about pages being in a particular page miss or page empty or page hit state.

4. (Original) The method claimed in claim 1, wherein a selected write may be preempted by a read to optimize latency.

5. (Original) The method claimed in claim 1, further comprising:

selectively scheduling DRAM commands at intervals longer than minimum DRAM timings core timings to accommodate higher density of command scheduling achieved from out of order scheduling.

6. (Original) The method claimed in claim 1, wherein there is a queue for each bank and rank combination.

7. (Currently Amended) The method claimed in claim 1, wherein within at least one queue, sorting memory transactions by attributes further comprises:

sorting memory transactions into queues in response to attributes ~~such as~~including address and address mapping used by memory, processor priority and demand versus prefetch transaction type.

8. (Original) The method claimed in claim 1, further comprising:

determining globally oldest unblocked memory transaction and locally oldest unblocked memory transaction for each queue;

determining whether last scheduled read was a page hit and more unblocked read page hits are queued to that same page, and either consecutive number of page hits serviced from that page is less than a predefined number or no reads pending to other banks.

9. (Original) The method claimed in claim 8, further comprising:

in response to absence of a globally oldest unblocked read that is a page hit, last scheduled read that is a page hit, oldest of the locally oldest unblocked page hit and oldest of the locally oldest page empty existing, scheduling a globally oldest unblocked read page miss.

10. (Original) The method claimed in claim 8, further comprising

in response to a scheduled page hit and a last scheduled read that is also a page hit and both are to the same page, incrementing a count, else setting the number of consecutive hits to zero;

in response to no consecutive page hits selected, resetting count to zero;

resetting the last scheduled read to hit in response to read queues being empty for preset duration; and

returning to start of the scheduling cycle.

11. (Original) The method claimed in claim 8, wherein the globally oldest unblocked memory transaction and locally oldest unblocked memory transaction for each queue may be a read hit, empty or miss.

12. (Original) The method claimed in claim 8, wherein determining whether last scheduled read was a page hit and more unblocked read page hits are queued to that same page, and either consecutive number of page hits serviced from that page is less than a predefined number or no reads pending to other banks further comprises:

scheduling an unblocked page hit;

in response to a scheduled page hit and a last scheduled read that is also a page hit and both are to the same page, incrementing the count else setting the number of consecutive hits to zero;

in response to no consecutive page hits selected, resetting the count to zero;

resetting the last scheduled read to hit in response to read queues being empty for preset duration; and

returning to start of the scheduling cycle.

13. (Original) The method claimed in claim 8, wherein in response to a globally oldest unblocked read being a page hit, scheduling a globally oldest unblocked page hit;

in response to a scheduled page hit and a last scheduled read that is also a page hit and both are to the same page, incrementing the count else setting the number of consecutive hits to zero;

in response to no consecutive page hits selected, resetting the count to zero;

resetting the last scheduled read to hit in response to read queues being empty for preset duration; and

returning to start of the scheduling cycle.

14. (Original) The method claimed in claim 8, wherein determining whether the last scheduled read is a page hit further comprises:

scheduling the globally oldest unblocked read;

in response to a scheduled page hit and a last scheduled read that is also a page hit and both are to the same page, incrementing the count else set the number of consecutive hits to zero;

in response to no consecutive page hits selected, resetting the count to zero;

resetting the last scheduled read to hit in response to read queues being empty for preset duration; and

returning to start of the scheduling cycle.

15. (Original) The method claimed in claim 8, wherein determining whether the oldest of the locally oldest unblocked page hit exists further comprises:

scheduling the oldest of the locally oldest unblocked page hit;

in response to a scheduled page hit and a last scheduled read that is also a page hit and both are to the same page, incrementing the count else set the number of consecutive hits to zero;

in response to no consecutive page hits selected, resetting count to zero;

resetting the last scheduled read to hit in response to read queues being empty for preset duration; and

returning to start of the scheduling cycle.

16. (Original) The method claimed in claim 8, wherein determining whether the oldest of the locally oldest page empty exists further comprises:

scheduling the oldest of the locally oldest unblocked page empty;

in response to a scheduled page hit and a last scheduled read that is also a page hit and both are to the same page, incrementing the count else set the number of consecutive hits to zero;

in response to no consecutive page hits selected, resetting count to zero;

resetting the last scheduled read to hit in response to read queues being empty for preset duration; and

returning to start of the scheduling cycle.

17. (Original) The method claimed in claim 1, wherein scheduling memory transactions out of order in accordance with scheduling algorithms, further comprises:

in response to absence of an unblocked previously scheduled page miss or page empty, last scheduled read that is a page hit and non-conflicting unblocked read, scheduling an oldest unblocked read transaction.

18. (Original) The method claimed in claim 17, further comprising:

in response to an unblocked previously scheduled page miss or page empty, scheduling a next DRAM command for the oldest page miss or page empty.

19. (Original) The method claimed in claim 17, further comprising:

in response to a last scheduled read being a page hit, scheduling an oldest unblocked transaction and updating a last scheduled read.

20. (Original) The method claimed in claim 17, further comprising:

in response to the last scheduled read being a page hit, scheduling an oldest non-conflicting unblocked read transaction and updating a last scheduled read.

21. (Original) The method claimed in claim 1, wherein scheduling memory transactions out of order in accordance with scheduling algorithms, further comprises:

in response to absence of a globally oldest unblocked page hit and globally oldest unblocked page empty, scheduling an oldest unblocked read transaction.

22. (Original) The method claimed in claim 21, further comprising:

in response to a globally oldest unblocked page hit, scheduling the oldest unblocked page hit.

23. (Original) The method claimed in claim 21 further comprising:

in response to globally oldest unblocked page empty, scheduling the oldest unblocked page empty.

24 (Original) The method claimed in claim 1, wherein scheduling memory transactions out of order in accordance with scheduling algorithms, further comprises:

in response to absence of an unblocked write that is on-page to last scheduled write and non-conflicting unblocked write, scheduling the oldest unblocked write transaction and updating the last scheduled write.

25. (Original) The method claimed in claim 24, further comprising:

in response to an unblocked write that is on-page to a last scheduled write, scheduling an unblocked on-page write and updating the last scheduled write.

26. (Original) The method claimed in claim 24, further comprising:

in response to non-conflicting unblocked write, scheduling selected non-conflicting unblocked write transaction and updating the last scheduled write.

27. (Original) The method claimed in claim 1, wherein scheduling memory transactions out of order in accordance with scheduling algorithms, further comprises:

in response to absence of a last scheduled write that is a page hit and a non-conflicting unblocked write, scheduling the oldest unblocked write transaction and updating the last scheduled write.

28. (Original) The method claimed in claim 27, further comprising:

in response to a last scheduled write is a page hit, scheduling the oldest unblocked transaction and updating the last scheduled write.

29. (Original) The method claimed in claim 27, further comprising:

in response to a non-conflicting unblocked write, scheduling the oldest non-conflicting unblocked write transaction and updating the last scheduled write.

30. (Original) A machine readable medium having stored therein a plurality of machine readable instructions executable by a processor for out of order memory scheduling using at least one queue, comprising:

instructions to sort memory transactions into at least one queue;

instructions to, within at least one queue, sorting memory transactions by attributes;

instructions to schedule memory transactions out of order in accordance with scheduling algorithms, wherein if a DRAM command of a later transaction is scheduled before the last DRAM command of an earlier transaction, the later transaction is considered scheduled out of order and wherein the entire at least one queue is considered while making scheduling decisions; and

instructions to make out of order selection at the time of DRAM command launch.

31. (Original) The machine readable medium claimed in claim 30, wherein making out of order selection at the time of DRAM command launch further comprises:

instructions to, in response to making an out of order selection at the time of DRAM command launch and not earlier, allow at least one queue to fill up without delaying the DRAM command.

32. (Original) The machine readable medium claimed in claim 30, further comprising:

instructions to accommodate page closing policy with no prior assumption being made about pages being in a particular page miss or page empty or page hit state.

33. (Original) The machine readable medium claimed in claim 30, further comprising:

instructions to preempt a selected write transaction by a read transaction to optimize latency.

34. (Original) The machine readable medium claimed in claim 30, further comprising:

instructions to selectively schedule DRAM commands at intervals longer than minimum DRAM timings core timings to accommodate higher density of command scheduling achieved from out of order scheduling.

35. (Original) The machine readable medium claimed in claim 30, wherein there is a queue for each bank and rank combination.

36. (Currently Amended) The machine readable medium claimed in claim 30, wherein instructions to within at least one queue, sort memory transactions by attributes further comprises:

instructions to sort memory transactions into queues in response to attributes ~~such as~~including read address and address mapping used by memory, processor priority and demand versus prefetch transaction type.

37. (Original) An apparatus comprising:

at least one queue associated with a memory bank for receiving memory transactions sorted by attributes;

a page table to store information related to memory transactions; and

a comparator and selector for receiving unblocked memory transactions from at least one queue and receiving page table information to determined if the unblocked transaction is a page hit, page miss or page empty,

wherein memory transactions are scheduled out of order in accordance with scheduling algorithms, the entire at least one queue is considered while making scheduling decisions and out of order selections are made at the time of DRAM command launch.

38. (Original) The apparatus claimed in claim 37, wherein if a DRAM command of a later memory transaction is scheduled before the last DRAM command of an earlier memory transaction, the later memory transaction is considered scheduled out of order.

39. (Original) The apparatus claimed in claim 37, further comprising:

an arbitrator for prioritizing between read and write memory transactions.

40. (Original) The apparatus claimed in claim 37, further comprising:

a content associative memory (CAM) to determine if memory transactions are page hits or misses.

41. (Original) The apparatus claimed in claim 40, wherein the memory transactions comprise write transactions.

42. (Original) The apparatus claimed in claim 37, wherein in response to making an out of order selection at the time of DRAM command launch, at least one queue fills up without delaying the DRAM command.

43. (Original) The apparatus claimed in claim 37, wherein page closing policy accommodations are made with no prior assumption about pages being in a particular page miss or page empty or page hit state.

44. (Original) The apparatus claimed in claim 37, DRAM commands are selectively scheduled at intervals longer than minimum DRAM timings core timings to accommodate higher density of command scheduling achieved from out of order scheduling.

45. (Original) The apparatus claimed in claim 37, wherein there is a queue for each bank and rank combination.

46. (Currently Amended) The apparatus claimed in claim 37, wherein within at least one queue, memory transactions are sorted into at least one queue in response to attributes ~~such as~~including address and address mapping used by memory, processor priority and demand versus prefetch transaction type.

47. (Original) A system comprising:

a memory including a plurality of banks; and

a memory controller operatively in communication with the memory;

at least one queue associated with a memory bank for receiving memory transactions sorted by attributes;

a page table to store information related to memory transactions; and

a comparator and selector for receiving unblocked memory transactions from at least one queue and receiving page table information to determined if the unblocked transaction is a page hit, page miss or page empty,

wherein memory transactions are scheduled out of order in accordance with scheduling algorithms, the entire at least one queue is considered while making scheduling decisions and out of order selections are made at the time of DRAM command launch.

48. (Original) The system claimed in claim 47, wherein if a DRAM command of a later memory transaction is scheduled before the last DRAM command of an earlier memory transaction, the later memory transaction is considered scheduled out of order.

Claims 49-70 (Canceled).